Project Design Report

Online Banking System

[SNAPS]

[11/05/2019]

Anusha Palisetty

Chaitanya Sai Kumar Talluru

MOHAMMED SHOEBUDDIN HABEEB

Naveen Kumar Vadlakonda

Pragna Reddy Kancharla

Introduction

As part of this project we would be introducing how customer account information maintenance and management is achieved through database in Banking applications. The main goal is to develop a database that would be apt and provide an efficient means of data management in banking domain.

Throughout the globe, Financial services plays a vital role in transactions between producers and end users. This application will provide a platform to view and manage the financial transactions of the user related to bank. It helps to keep track of day to day user financial statistics and allows the user to manage the accounts digitally.

This database bridges the gap between customers and banking sector, where we would be able to maintain the bank and customer related data which can further be used in banking or web applications which supports multiple functionalities like Online Banking, Account Management, transaction initiations by customers or bank agents, Online payments and many or which would be depicted in ER diagram.

Requirements Analysis

Data Requirements

Our database consists of total 27 tables and each table is identified with that tables primary key. Below are the main table names for our data base design. We also define Foreign keys based on the relationship between the entities.

Functional Requirements

- The system shall provide an option for the customer to view the accounts and the corresponding balances.
- The system shall provide an option for the customer to view the transactional details.
- The system shall provide an option to transfer the money from accounts.
- The system shall provide an option to manage the customer's personal details.
- The system shall provide the database for customer's different cards information.
- The system shall provide the database for customer's different types of accounts like Savings, Checking, IRA, etc.
- The system shall provide the database for customer's different types of loans like Personal, Student, Mortgage and Home loan.
- The system shall provide the database which shows the relationship between banker and customer.
- The system shall validate the amount available in the user's account before releasing funds

to the user.

- The system shall update the user's account upon transfer of funds.

Conceptual Design

Entities and Attributes

1.Account: Describes the type of the account that the customer holds in bank

Attributes

- Act.no: Account Number is the unique one for an account of the customer
- Category: Type of the account can be categorized into 5 types
- Balance: Specifies the amount in corresponding account.
- Cust_Id: Specifies Customer Id

Relationships

- Relationship1: Customer has an account
- Relationship2: Branch manages Account

Primary Key

- It is identified by AcctNo attribute, because Account Number is unique for any type of account.

Foreign Key

- It is identified by Cust_Id attribute, References back to the Cust_Id of customer table

2.Bank: Describes the various attributes of the bank

Attributes

- Bank_Name: Specifies the name of the bank which is unique.
- Bank_Headquaters: Specifies the headquarter address of the bank.

Relationships

R1: Bank owns many Branches

Primary Key

- It is identified by Bank_Name attribute, because every bank has the unique name.

3.Branch: Describes the branches of the bank

Attributes

- Branch_Code: Uniquely identifies each branch.
- Branch_Name: Name of the branch.
- Bank_Name: Specifies the name of the bank to which the branch belongs.
- Manager: Specifies the manager of the branch.
- Assets: Specifies the assets of each branch.

Relationships

- R1: Bank owns many branches.
- R2: Employee works at branch

Primary Key

- It is identified by Branch_Code attribute, because it is the unique code for each branch.

Foreign Key

Bank_Name References back to the Bank_Name of bank table

4.Employee (Banker): Describes the details of the bank employees

Attributes

- Branker_Id: Uniquely identifies each employee of the bank.
- Banker_Name: Specifies the name of the banker.
- Branch_Code: Specifies the branch in which banker works.
- Address: Specifies Address of the employee.
- Banker_email: Email address of the banker.

Relationships

- R1: Employee works at branch.
- R2: Customer banks with banker.

Primary Key

It is identified by Banker_Id attribute, because it is the unique code for each employee.

Foreign Key

Branch_Code References back to the Branch_Code of branch table.

5.Loans: Describes the loans of the customer

Attributes

- Loan_id: Uniquely identifies loan.
- Loan_Amount: Specifies the amount of the loan.

Relationships

Relationship1: Customer borrows loan.

Primary Key

It is identified by Loan_Id attribute, because it is the unique code for each loan.

6. Personal Loan: Describes the personal loan of the customer

Attributes

- Personal_Loan_id: Uniquely identifies loan.
- Customer_No: Specifies the receiver of the loan.
- Employer_Name: Specifies name of the company that person is working in
- Employee_Id: Specifies the employee id
- DateLoanGiven: Specifies the date of the loan sanctioned.
- AmountGiven: Specifies the amount of the loan.
- InterestRate: Loan interest is specified by this attribute.

Relationships

Relationship1: Customer borrows loan

Primary Key

It is identified by a combination of Employee_ID and Employer_name, because it is the unique code for each loan.

Foreign Key

Loan_id References back to the Loan_id Loan table.

7. Home Loan: Describes the home loan of the customer

Attributes

- Loan_id: Uniquely identifies loan.
- Customer_No: Specifies the receiver of the loan.
- Sale_deed: specifies the registration id of the assert
- DateLoanGiven: Specifies the date of the loan sanctioned.
- AmountGiven: Specifies the amount of the loan.
- InterestRate: Loan interest is specified by this attribute.

Relationships

Relationship1: Customer borrows loan

Primary Key

It is identified by Sale_deed attribute, because it is the unique code for each loan.

Foreign Key

Loan_id References back to the Loan_id Loan table.

8.Student Loan: Describes the student loan of the customer

Attributes

- Loan_id: Uniquely identifies loan.
- Customer_No: Specifies the receiver of the loan.
- Student_Id: Specifies the student id
- Student_mail: Specifies student mail
- DateLoanGiven: Specifies the date of the loan sanctioned.
- AmountGiven: Specifies the amount of the loan.
- InterestRate: Loan interest is specified by this attribute.

Relationships

Relationship1: Customer borrows loan

Primary Key

It is identified by Student_Id and Student_mail attribute, because it is the unique code for each loan.

Foreign Key

Loan_id References back to the Loan_id Loan table.

9.Vehicle Loan: Describes the student loan of the customer

Attributes

- Loan_id: Uniquely identifies loan.
- Customer_No: Specifies the receiver of the loan.
- Chassis_No: Uniquely identifies the vehicle.
- DateLoanGiven: Specifies the date of the loan sanctioned.
- AmountGiven: Specifies the amount of the loan.
- InterestRate: Loan interest is specified by this attribute.

Relationships

Relationship1: Customer borrows loan.

Primary Key

It is identified by Chasis_No attribute, because it is the unique code for each loan.

Foreign Key

Loan_id References back to the Loan_id Loan table.

10.Cards: Describes the different cards that customer holds.

Attributes

- Customer_No: Foreign Key references customer.
- Card_No: Unique number of card

Relationships

Relationship1: cards are linked to customer

Primary Key

It is identified by Card_No attribute, because it is unique for each card.

Foreign Key

Customer_No References back to the Customer_No of customer table

11.CreditCards: Describes the credit card details of the customer.

Attributes

- NameOnCard: Describes the name on the card.
- Card_No: Specifies the card number

- SecurityCode: Specifies the Security number
- ExpMonth: Specifies the expiry month
- ExpYear: Specifies the expiry year
- BillingPostalCode: Specifies the postal code for billing
- AMOUNT: Specifies the balance of the card
- Cardlimit: Specifies the maximum limit of the card
- IsActive: Checks whether card is active or not

Relationships

Relationship1: Customer has a card.

Primary Key

It is identified by Card_No attribute, because it is unique code for each card.

Foreign Key

Card_No References back to the Card_No of Cards table

12.DebitCards: Describes the debit card details of the customer.

Attributes

- NameOnCard: Describes the name on the card
- Card_No: Specifies the card number
- SecurityCode: Specifies the Security number
- ExpMonth: Specifies the expiry month
- ExpYear: Specifies the expiry year
- Balance: Specifies the balance of the card
- IsActive: Checks whether card is active or not

Relationships

Relationship1: Customer has a card.

Primary Key

It is identified by Card_No attribute, because it is unique code for each card.

Foreign Key

Card_No References back to the Card_No of Cards table

13.TravelCurrencyCard: Describes the travel card details of the customer.

Attributes

- NameOnCard: Describes the name on the card
- Card_No: Specifies the card number
- SecurityCode: Specifies the Security number
- ExpMonth: Specifies the expiry month
- ExpYear: Specifies the expiry year
- AMOUNT: Specifies the balance of the card
- IsActive: Checks whether card is active or not

Relationships

Relationship1: Customer has a card.

Primary Key

It is identified by Card_No attribute, because it is unique code for each card.

Foreign Key

Card_No References back to the Card_No of Cards table

14.EMICard: Describes the travel card details of the customer.

Attributes

- NameOnCard: Describes the name on the card
- Card_No: Specifies the card number
- SecurityCode: Specifies the Security number
- ExpMonth: Specifies the expiry month
- ExpYear: Specifies the expiry year
- EMIAmount: Specifies the EMI amount.
- IsActive: Checks whether card is active or not

Relationships

Relationship1: Customer has a card.

Primary Key

It is identified by Card_No attribute, because it is unique code for each card.

Foreign Key

Card_No References back to the Card_No of Cards table

15.LoyalityCard: Describes the travel card details of the customer.

Attributes

- NameOnCard: Describes the name on the card
- Card_No: Specifies the card number
- SecurityCode: Specifies the Security number
- ExpMonth: Specifies the expiry month
- ExpYear: Specifies the expiry year
- Points: Specifies the points earned
- Amount: Specifies the amount present
- IsActive: Checks whether card is active or not

Relationships

Relationship1: Customer has a card.

Primary Key

It is identified by Card_No attribute, because it is unique code for each card.

Foreign Key

Card_No References back to the Card_No of Cards table

16.CorporateCard: Describes the travel card details of the customer.

Attributes

- NameOnCard: Describes the name on the card
- Card_No: Specifies the card number
- SecurityCode: Specifies the Security number
- ExpMonth: Specifies the expiry month
- ExpYear: Specifies the expiry year
- Amount: Specifies the amount present

- IsActive: Checks whether card is active or not

Relationships

Relationship1: Customer has a card.

Primary Key

It is identified by Card_No attribute, because it is unique code for each card.

Foreign Key

Card_No References back to the Card_No of Cards table

17: SavingsAccount

Attributes

- AccountNo: Specifies the customer AccountNo.
- AccountOpenDate: Specifies Account Open date
- AccountCloseDate: Specifies Account Close date
- AccountMaturityDate: Specifies Maturity date
- AvailableBalance : Specifies the available balance
- InterestsEarned : Specifies the interest amount earned
- Apr: Specifies the annual price rate

Relationships

Customer has an account

Primary Key

- AccountNo is identified as unique primary number

Foreign Key

- AccounNo References back to the AcctNo of *Account* table

18: CertificateOf Deposit

Attributes

- AccountNo: Specifies the customer AccountNo.
- AccountOpenDate: Specifies Account Open date
- AccountCloseDate: Specifies Account Close date
- AccountMaturityDate: Specifies Maturity date

- AvailableBalance : Specifies the available balance
- InterestsEarned : Specifies the interest amount earned
- Apr: Specifies the annual price rate
- InteresType: Specifies fixed or variable
- EarlyWithdrawl: Specifies penalty fees

Relationships

Customer has an account

Primary Key

- AccountNo is identified as unique primary number

Foreign Key

- AccounNo References back to the AcctNo of *Account* table

19 : CheckingsAccount

Attributes

- AccountNo: Specifies the customer AccountNo.
- AccountOpenDate: Specifies Account Open date
- AccountCloseDate: Specifies Account Close date
- AccountMaturityDate: Specifies Maturity date
- AvailableBalance : Specifies the available balance
- InterestsEarned : Specifies the interest amount earned
- Apr: Specifies the annual price rate

Relationships

Customer has an account

Primary Key

- AccountNo is identified as unique primary number

Foreign Key

- AccounNo References back to the AcctNo of *Account* table

20 : MoneyMarketAccount

Attributes

- AccountNo: Specifies the customer AccountNo.
- AccountOpenDate: Specifies Account Open date
- AccountCloseDate: Specifies Account Close date
- AccountMaturityDate: Specifies Maturity date
- AvailableBalance : Specifies the available balance
- InterestsEarned : Specifies the interest amount earned
- Apr: Specifies the annual price rate
- TransactionLimit:Number of transactions can be performed in a month

Relationships

Customer has an account

Primary Key

- AccountNo is identified as unique primary number

Foreign Key

- AccounNo References back to the AcctNo of *Account* table

21 : IRAAccount

Attributes

- AccountNo: Specifies the customer AccountNo.
- AccountOpenDate: Specifies Account Open date
- AccountCloseDate: Specifies Account Close date
- AccountMaturityDate: Specifies Maturity date
- AvailableBalance : Specifies the available balance
- InterestsEarned : Specifies the interest amount earned
- Apr: Specifies the annual price rate
- IRAType:Specifies IRA type
- Age: Specifies age limit for account.

Relationships

Customer has an account

Primary Key

- AccountNo is identified as unique primary number

Foreign Key

- AccounNo References back to the AcctNo of *Account* table

22: CorporateAccount

Attributes

- AccountNo: Specifies the customer AccountNo.
- AccountOpenDate: Specifies Account Open date
- AccountCloseDate: Specifies Account Close date
- AccountMaturityDate: Specifies Maturity date
- AvailableBalance : Specifies the available balance
- InterestsEarned : Specifies the interest amount earned
- Apr: Specifies the annual price rate
- RecurringDeposit: yes or no

Relationships

Customer has an account

Primary Key

- AccountNo is identified as unique primary number

Foreign Key

- AccounNo References back to the AcctNo of *Account* table

23: Customer

Customer: This table contains the personal details of the customer

Attributes

- Cust_Id: ID of a customer
- Cust_Name: Name of the customer
- Cust_Dob: Date of birth of customer

- SSN: Identity Proof of customer
- Cust_Occ : Occupation of customer
- Cust _Email : Email Id of customer
- Cust_PhoneNo: Phone number of the customer
- Banker_Id : Specifies the banker he banks with.

Relationships

This is a customer table. Each account has a customer. Account to customer has (1,1) cardinality and customer to account has (1,20)

Primary Key

- Cust_Id is identified as unique primary number

Foreign Key

- Banker_Id References back to the Banker_Id of *Banker* table

24: Transactions

Attributes

- Trans_Id: Transaction ID of a account
- Trans_Type: If its digital or non-digital
- Trans_Timestamp: Date of birth of customer
- Tran_Status: Transaction success or not.
- Trans_Amt : Occupation of customer
- Account_Number : Account Number from which transaction is done
- Trans_Name : Purpose of transaction

Relationships

Account does the transactions. Account to transactions cardinality is (0,N) and transactions to account has (1,1)

Primary Key

- Trans_ID is identified as unique primary number

25: Alert

Attributes

- Alert_Id: Alert_ID of customer
- Alert_Type: Type of alert like SMS, email, mail
- Alert_Timestamp: Time the alert is sent
- Cust_ID: Customer ID to which alerts will be sent
- Alert_Status : Status of Alert

Relationships

Alerts are sent to every customer based on their activities. Customer to Alert has (0,N) and Alert to customer has (1,N)

Primary Key

- Alert_ID is identified as unique primary number

Foreign Key

Cust_ID References back to the Customer_Id of *Customer* table

26: Penalties

Attributes

- Penalty_Id: Alert_ID of customer
- Acct_No: Type of alert like SMS, email, mail
- Penalty_Status: Status of Penalty
- Penalty_Amt: Customer ID to which alerts will be sent
- Charged_Date: Date the penalty is applied

Relationships

Each account is charged with Penalty if it does not meet minimum requirement of the bank. Account to Penalty has (0,N) cardinality and Penalty to account has (1,1) cardinality.

Primary Key

- Penalty_ID is identified as unique primary number

Foreign Key

Acc_No References back to the Account_No of *Accounts* table

27: Benefits

Attributes

- Benefit_Id: Unique ID of the benefits provided to the customer
- Benefit_Type: Like Reward points, Gift cards, Cash back amount
- Cust_ID: Customer ID to which the benefit goes to.
- Benefit_Amount : The amount with respect to the benefit
- Benefit_Dt : Date on which the benefit is provided to the customer.

Relationships

Customer is provided benefits. Customer to benefit has (0,N) cardinality and Benefit to customer has (1,N)

Primary Key

- Benefit_Id: It is identified as unique primary number.

Foreign Key

Cust_ID References back to the Customer_Id of *Customer* table

EER Model:

This ER diagram represents the model of Banking Management System. The Entity relationship of the banking management system shows the visual instrument of database tables and relations between Account, Customers, Transactions, Cards, Loans. It is used to structure the data and to define the relationships between structured data groups of Banking Management System functionalities. Cardinality for each relationship is provided in the ER diagram and the maxium cardinality ratio for our design is 1:10. ISA relationship exists for Accounts, Loans and Cards entities. Specialization property is applied on the ISA relationships i.e a top-down design where new subclasses are created from the existing class.

- Loans have Home, Student, Personal,Vehicle loans as sub-classes.
- Cards have Loyalty, Debit,Credit, TravelCurrency, Corporate,EMI as sub-classes.
- Accounts have Checkings,Savings, MoneyMarket, CertificateOfDeposit,IRA,Corporate as sub-classes.

The Cards linked by Customer, Account for Customer, Customer serviced by Banker, Transaction made by Account has 1:1 relationship. All the primary keys for each entity are mentioned as underlined coulumn in the ER diagram. The attributes description and the

primary keys for each entity are mentioned in the section Entities and Attributes .

Relational Model:

A Relational model is built based on the relational approach between entities in the E-R diagram. The Primary Key, Foreign Key specifications are provided in the relational model. The primary key attributes are underlined which uniquely identify the record. For our Banking database system we have total 17 relations between the entities. The primary key attributes PK of each relation schema cannot have null values in any tuples. A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table. In our relational model we mapped CUST_ID from Cards to CUST_ID of Customer. Here CUST_ID(Cards) acts as a Foreign key where CUST_ID(Customer) is the primary key. Similarly Foreign Keys are defined for all the relations in our banking system relational model.

For Specialization mappings like Account, Loans, Cards having multiple relation the relational model is designed based on the ER model.

- The Account which has single relation with multiple attributes is merged into one relation with Acct_Type column differentiating the type of the Account.
- The Card which has single relation with multiple attributes is merged into one relation with Card_Type column differentiating the type of the Card.
- The Loans which has different primary keys for the subclasses have separate relations for each subclass and all the Loan_ID's from the sub-classes have direct arc to Loans master table.
- For the (1,N) relations for Benefits and Alerts two relation USES_BENEFITS, GETS_ALERT relations have been implemented.

All the relations created already have 1NF. The 2NF and 3NF are implemented for based on this relational mapping. All the attributes are made sure they are dependent on the primary key and no non-prime attribute has transitive dependency on the primary key. For the CARDS relation all the attributes dependent on the CARD_NO are included and the remaining attributes we split into CARDS_CUSTOMER which contains the attributes dependent on both CUST_ID and CARD_NO. Similarly for the Personal Loan the relation is split based on the functional dependency and 2NF. All the attributes which are dependent on Employer Name and Employee ID are in one relation and remaining attributes which dependends on both Personal_Loan_Id and Cust_Id have another relation. There no much cases where 3NF can be implemented as all the non-prime attributes are diretly dependent on the primary key.

Data Dictionary:

A data dictionary is created which contains the meta data of all the tables. It provides the table names, attributes, data type, primary key, foreign key and constraints for each attribute. For all the Acct_No,Cust_id,card_No Bigint data type is used. Decimal data type is used for all the balances related attributes. ENUM data type is used for where ever 'Y' or 'N' type data is used. Detailed information on the data dictionary are provided in the attachment.

Implementation:

Based on the Relational model tables were created using CREATE statements and primary key and foreign key constraints were added accordingly. The queries which explains the functional requirement of our Banking system are provided in the SQL queries document.

- The query which views the account for a particular customer.
- The query which shows the transaction details of a customer.
- The query which provides the Customer personal details.
- The query for a Customer and his different account types , loan types.
- The query which provides the Banker_ID for a customer.
- The query which shows the Users Account Balance before and after transaction.

Summary:

A Banking System Database is designed for a banking application. All the phases for this databases are implemented as part of this project and UI is also built for the same. All the functional requirements are implemented in the databases which helps the banking applications for easy data management. ER, Functional dependency, Normalization, SQL, UI concepts are covered as part of this project.

Teamwork

Anusha,Soheb : Worked on the ER, Relational Mapping, Functional Dependency, Normalization, SQL.

Naveen,Chaitanya,Pragna :

- Outline of ER,
- conceptual design of ER
- SQL updating, insertion, deletion as per UI design.
- UI front end designed using PHP
- HTML Validations
- Java Script
- WAMP server
- MYSQL database

Appendix:

- We are unable to add the ER model into the document.So, please refer to separate file name Final ER.pdf

- We are unable to add the Final relational mapping here so, please refer to Relational_Mapping_Final.pdf